

On the Computational Complexity of Vertex Integrity^{*}

Pål Grønås Drange, Markus Sortland Dregi, and Pim van 't Hof

Department of Informatics, University of Bergen, Norway,
`{pal.drange,markus.dregi,pim.vanthof}@ii.uib.no`

Abstract. We study two different but closely related vulnerability measures of graphs, namely *vertex integrity* and *component order connectivity*. Both parameters take into account not only the number of vertices that need to be deleted in order to break a graph into pieces, but also the number of vertices in the largest connected component that remains. Kratsch et al. (Discr. Appl. Math. 77: 259–270, 1997) presented polynomial-time algorithms for computing the vertex integrity for various classes of intersection graphs. In particular, they showed that this problem can be solved in $O(n^{2d+1})$ time on co-comparability graphs of dimension at most d , provided that an intersection model of the input graph is given as part of the input. The weighted version of the problem is known to be solvable in polynomial time on interval graphs, but is NP-complete on chordal graphs. We present several new classical and parameterized complexity results for the weighted and unweighted versions of the problem. In particular, we show that determining the (unweighted) vertex integrity is NP-complete on co-comparability graphs, thereby refuting a conjecture by Ray and Deogun (J. Combin. Math. Combin. Comput. 16: 65–73, 1994). We also obtain several classical and parameterized complexity results involving the component order connectivity, uncovering interesting similarities and differences between the two vulnerability measures.

1 Introduction

Motivated by a multitude of practical applications, many different vulnerability measures of graphs have been introduced in the literature over the past few decades. The vertex and edge connectivity of a graph, although undoubtedly being the most well-studied of these measures, often fail to capture the more subtle vulnerability properties of networks that one might wish to consider, such as the number of resulting components, the size of the largest or smallest component, and the largest difference in size between any two components. The two vulnerability measures we study in this paper, *vertex integrity* and *component order connectivity*, take into account not only the number of vertices that need to be deleted in order to break a graph into pieces, but also the number of vertices in the largest component that remains.

The vertex integrity of a graph is a vulnerability measure that was introduced by Barefoot, Entringer and Swart [2] in 1987. A more general weighted variant of the measure was defined by Ray and Deogun [18] a few years later. In order to formally define vertex integrity, we need to introduce some terminology. Let G be a graph and $w : V(G) \rightarrow \mathbb{N} =$

^{*} The research leading to these results has received funding from the Research Council of Norway and the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 267959.

$\{0, 1, \dots\}$ a weight function on the vertices of G . The weight of a subset $X \subseteq V(G)$ is defined as $w(X) = \sum_{v \in X} w(v)$. For our purposes, it is convenient to define the weight of G , denoted by $w(G)$, as follows. If G is connected, then $w(G)$ is the weight of its vertex set. If G is disconnected, then $w(G)$ is the weight of a *heaviest component* of G , that is, $w(G) = w(D)$ for a component D whose weight is at least as high as the weight of any other component of G . The *vertex integrity* of G is defined as

$$\iota(G) = \min\{w(X) + w(G - X) \mid X \subseteq V(G)\},$$

where $G - X$ denotes the graph obtained from G by deleting all the vertices in X . Any set $X \subseteq V(G)$ for which $w(X) + w(G - X) = \iota(G)$ is called an ι -set of G . Since an unweighted graph can be interpreted as a graph whose vertices all have weight 1, the vertex integrity of an unweighted graph G equals $\iota(G) = \min\{|X| + n(G - X) \mid X \subseteq V(G)\}$, where $n(G - X)$ denotes the number of vertices in a largest component of $G - X$.

For an overview of several structural results on vertex integrity, including combinatorial bounds and relationships between vertex integrity and other vulnerability measures, we refer the reader to a survey on the subject by Bagga et al. [1]. We mention here only results on the computational complexity of determining the (weighted) vertex integrity of a graph.

The VERTEX INTEGRITY problem takes as input a graph G and an integer p , and asks whether $\iota(G) \leq p$. This problem was shown to be NP-complete, even when restricted to planar graphs, by Clark, Entringer, and Fellows [6]. On the positive side, Fellows and Stueckle [9] showed that the problem can be solved in $O(p^3 n)$ time, and is thus fixed-parameter tractable when parameterized by p . In the aforementioned survey, Bagga et al. [1] mention that the unweighted VERTEX INTEGRITY problem can be solved in $O(n^3)$ time when the input graph is a tree or a cactus graph. Kratsch, Kloks and Müller [15] studied the computational complexity of determining the value of several different vulnerability measures in classes of intersection graphs. Their results imply that VERTEX INTEGRITY can be solved in $O(n^3)$ time on interval graphs, in $O(n^4)$ time on circular-arc graphs, and in $O(n^5)$ time on permutation graphs and trapezoid graphs. Kratsch et al. [15] also mention that the problem can be solved in $O(n^{2d+1})$ time on co-comparability graphs of dimension at most d , provided that an intersection model of the input graph is given as part of the input.

Ray and Deogun [18] were the first to study the more difficult WEIGHTED VERTEX INTEGRITY problem. Using a reduction from 0-1 KNAPSACK, they identified several graph classes on which the WEIGHTED VERTEX INTEGRITY problem remains NP-complete. In particular, their result implies that the problem is NP-complete on graph classes such as trees, bipartite graphs, series-parallel graphs, and regular graphs, and therefore also on superclasses such as chordal graphs and comparability graphs. A common property of these classes is that they contain graphs with arbitrarily many asteroidal triples and induced paths of length 4; any graph class that does not have this property is not covered by the result Ray and Deogun. In fact, Ray and Deogun conjectured that the WEIGHTED VERTEX INTEGRITY problem can be solved in polynomial time on co-comparability graphs, a well-known example of a class of graphs that do not contain asteroidal triples at all. More than a decade later, Ray et al. [19] presented a polynomial-time algorithm for WEIGHTED VERTEX INTEGRITY on interval graphs, a subclass of co-comparability graphs. In the same

paper, they pointed out that the complexity of the problem on co-comparability graphs remains unknown.

We now turn our attention to the second vulnerability measure studied in this paper. For any positive integer ℓ , the ℓ -*component order connectivity* of a graph G is defined to be the cardinality of a smallest set $X \subseteq V(G)$ such that $n(G - X) < \ell$. We refer to the survey by Gross et al. [11] for more background, an overview of results, and references on component order connectivity.

Motivated by the definition of weighted vertex integrity, we consider the natural extension of component order connectivity to weighted graphs and introduce the following decision problem:

WEIGHTED COMPONENT ORDER CONNECTIVITY

Instance: A graph G , a weight function $w : V(G) \rightarrow \mathbb{N}$, and two non-negative integers k and ℓ .

Question: Is there a set $X \subseteq V(G)$ with $w(X) \leq k$ such that $w(G - X) \leq \ell$?

Any set $X \subseteq V(G)$ satisfying $w(X) \leq k$ and $w(G - X) \leq \ell$ is called a *solution* for the instance (G, w, k, ℓ) . The unweighted variant of the problem, i.e., the case where $w(v) = 1$ for every $v \in V(G)$, is called **COMPONENT ORDER CONNECTIVITY**.

In some sense, the **WEIGHTED COMPONENT ORDER CONNECTIVITY** problem can be seen as a more refined version of **WEIGHTED VERTEX INTEGRITY**. We therefore find it surprising that, to the best of our knowledge, the **WEIGHTED COMPONENT ORDER CONNECTIVITY** problem has not yet been studied in the literature. We do however point out that the techniques described by Kratsch et al. [15] yield polynomial-time algorithms for the unweighted version of the problem on interval graphs, circular-arc graphs, permutation graphs, and trapezoid graphs, and that very similar problems have received some attention recently [3, 11].

Our Contribution. In Section 2, we present our results on the weighted and unweighted versions of the **VERTEX INTEGRITY** problem. We show that **VERTEX INTEGRITY** is **NP**-complete on co-bipartite graphs, and hence on co-comparability graphs. This refutes the aforementioned conjecture by Ray and Deogun [18] and answers an open question by Ray et al. [19]. It also forms an interesting contrast with the result by Kratsch et al. [15] stating that **VERTEX INTEGRITY** can be solved in $O(n^{2d+1})$ time on co-comparability graphs of dimension at most d if an intersection model is given as part of the input. We also show that even though **VERTEX INTEGRITY** can be solved in linear time on split graphs, the problem remains **NP**-complete on chordal graphs. Interestingly, we prove that unlike the unweighted variant of the problem, the **WEIGHTED VERTEX INTEGRITY** problem is strongly **NP**-complete already on split graphs; observe that this does not follow from the aforementioned hardness result by Ray and Deogun [18], as split graphs do not contain induced paths of length 4.

Recall that Fellows and Stueckle [9] showed that **VERTEX INTEGRITY** can be solved in $O(p^{3p}n)$ time on general graphs. We show that the **WEIGHTED VERTEX INTEGRITY** problem can be solved in $O(p^{p+1}n)$ time, and that this problem admits a kernel with at most p^3 vertices, each having weight at most p .

Section 3 contains our results on the **WEIGHTED COMPONENT ORDER CONNECTIVITY** problem and its unweighted variant. The observation that there is a polynomial-time Turing reduction from **VERTEX INTEGRITY** to **COMPONENT ORDER CONNECTIVITY** implies

that the latter problem is NP-complete on any graph class for which VERTEX INTEGRITY is NP-complete. We prove that WEIGHTED COMPONENT ORDER CONNECTIVITY is NP-complete already on complete graphs, while the unweighted variant of the problem is NP-complete on split graphs. We find the latter result particularly interesting in light of existing polynomial-time algorithms for computing similar (unweighted) vulnerability measures of split graphs, such as toughness [20], vertex integrity, scattering number, tenacity, and rupture degree [16]. To complement our hardness results, we present a pseudo-polynomial-time algorithm that solves WEIGHTED COMPONENT ORDER CONNECTIVITY in $O(sn^3)$ time on interval graphs, where $s = \min\{k, \ell\}$. We then modify this algorithm to solve the unweighted version of the problem in $O(n^2)$ time on interval graphs, thereby improving the $O(n^3)$ -time algorithm that follows from the results by Kratsch et al. [15]. Observe that aforementioned hardness results rule out the possibility of solving WEIGHTED COMPONENT ORDER CONNECTIVITY in polynomial time on interval graphs or in pseudo-polynomial time on split graphs.

In Section 3.3, we completely classify the parameterized and kernelization complexity of the weighted and unweighted version of the COMPONENT ORDER CONNECTIVITY problem on general graphs with respect to the parameters k , ℓ , and $k + \ell$. Since COMPONENT ORDER CONNECTIVITY is equivalent to VERTEX COVER when $\ell = 1$, the unweighted problem is para-NP-hard when parameterized by ℓ . We prove that if we take k to be the parameter, then COMPONENT ORDER CONNECTIVITY is W[1]-hard even on split graphs. On the positive side, we show that WEIGHTED COMPONENT ORDER CONNECTIVITY becomes fixed-parameter tractable when parameterized by $k + \ell$. We present an algorithm for solving the problem in time $2^{O(k \log \ell)} n$, before proving that the problem cannot be solved in time $2^{o(k \log \ell)} n^{O(1)}$ unless the Exponential Time Hypothesis fails. Finally, we show that WEIGHTED COMPONENT ORDER CONNECTIVITY admits a kernel with at most $k\ell(k + \ell) + k$ vertices, where each vertex has weight at most $k + \ell$.

We refer to the monograph by Diestel [7] for graph terminology and notation not defined here. For more information on parameterized complexity and kernelization, we refer to [8].

2 Vertex Integrity

In Section 2.1, we present our classical complexity results on the WEIGHTED VERTEX INTEGRITY problem and its unweighted variant. The parameterized and kernelization complexity of the two problems will be discussed in Section 2.2.

2.1 Classical Complexity

As we mentioned in the introduction, Ray et al. [19] asked whether WEIGHTED VERTEX INTEGRITY can be solved in polynomial time on co-comparability graphs. We show that this is not the case, unless $P = NP$. In fact, we prove a much stronger result in Theorem 1 below by showing NP-completeness of an easier problem (VERTEX INTEGRITY) on a smaller graph class (co-bipartite graphs). To see why co-bipartite graphs form a subclass of co-comparability graphs, observe that every bipartite graph is a comparability graph, as directing all the edges of a bipartite graph from one bipartition class to the other yields a transitive orientation.

Theorem 1. VERTEX INTEGRITY is NP-complete on co-bipartite graphs.

Proof. The problem is clearly in NP. To show that it is NP-hard, we give a polynomial-time reduction from the BALANCED COMPLETE BIPARTITE SUBGRAPH problem. This problem, which is known to be NP-complete [10], takes as input a bipartite graph $G = (A, B, E)$ and an integer $k \geq 1$, and asks whether there exist subsets $A' \subseteq A$ and $B' \subseteq B$ such that $|A'| = |B'| = k$ and $G[A' \cup B']$ is a complete bipartite graph. Let (G, k) be an instance of BALANCED COMPLETE BIPARTITE SUBGRAPH, where $G = (A, B, E)$ is a bipartite graph on n vertices. We claim that (G, k) is a yes-instance of BALANCED COMPLETE BIPARTITE SUBGRAPH if and only if $(\overline{G}, n - k)$ is a yes-instance of VERTEX INTEGRITY.

Suppose there exist subsets $A' \subseteq A$ and $B' \subseteq B$ such that $|A'| = |B'| = k$ and $A' \cup B'$ induces a complete bipartite subgraph in G . Observe that in \overline{G} , both A' and B' are cliques, and there is no edge between A' and B' . Hence, if we delete all the vertices in $V(\overline{G}) \setminus (A' \cup B')$ from \overline{G} , the resulting graph has exactly two components containing exactly k vertices each. Since $|V(\overline{G}) \setminus (A' \cup B')| = n - 2k$, it holds that $\iota(\overline{G}) \leq n - 2k + k = n - k$, and hence $(\overline{G}, n - k)$ is a yes-instance of VERTEX INTEGRITY.

For the reverse direction, suppose $(\overline{G}, n - k)$ is a yes-instance of VERTEX INTEGRITY. Then there exists a subset $X \subseteq V(\overline{G})$ such that $|X| + n(\overline{G} - X) \leq n - k$. The assumption that $k \geq 1$ implies that $\overline{G} - X$ is disconnected, as otherwise $|X| + n(\overline{G} - X) = V(\overline{G}) = n$. Let $A' = A \setminus X$ and $B' = B \setminus X$. Since \overline{G} is co-bipartite, both A' and B' are cliques. Moreover, since $\overline{G} - X$ is disconnected, there is no edge between A' and B' . Hence, $\overline{G}[A']$ and $\overline{G}[B']$ are the two components of $\overline{G} - X$. Without loss of generality, suppose that $|A'| \geq |B'|$. Then $|B'| = n - (|X| + |A'|) = n - (|X| + n(\overline{G} - X)) \geq n - (n - k) = k$ and hence $|A'| \geq |B'| \geq k$. This, together with the observation that $A' \cup B'$ induces a complete bipartite subgraph in G , implies that (G, k) is a yes-instance of BALANCED COMPLETE BIPARTITE SUBGRAPH. \square

Ray and Deogun [18] proved that WEIGHTED VERTEX INTEGRITY is NP-complete on any graph class that satisfies certain conditions. Without explicitly stating these (rather technical) conditions here, let us point out that any graph class \mathcal{G} satisfying these conditions must contain graphs with arbitrarily many asteroidal triples and induced paths of length 4. Theorem 1 shows that neither of these two properties is necessary to ensure NP-completeness of WEIGHTED VERTEX INTEGRITY, since co-bipartite graphs contain neither asteroidal triples nor induced paths of length 4.

In Theorem 2 below, we show that WEIGHTED VERTEX INTEGRITY is NP-complete on split graphs. Since split graphs do not contain induced paths of length 4, this graph class is not covered by the aforementioned hardness result of Ray and Deogun [18]. We first prove two structural lemmas.

Lemma 1. For every graph G and weight function $w : V(G) \rightarrow \mathbb{N}$, there exists an ι -set X that contains no simplicial vertices of G .

Proof. Let $w : V(G) \rightarrow \mathbb{N}$ be a weight function of a graph G , and let X be an ι -set of G containing a simplicial vertex s . Observe that s is adjacent to at most one component of $G - X$. Let $X' = X \setminus \{s\}$. We claim that X' is an ι -set of G .

Let G_1, \dots, G_r denote the components of $G - X'$, and without loss of generality assume that $s \in V(G_1)$. The fact that s is a simplicial vertex of G implies that G_2, \dots, G_r are components of $G - X$ as well. Hence $w(G_i) \leq w(G - X)$ for every $i \in \{2, \dots, r\}$. Let

us determine an upper bound on $w(G_1)$. If s is adjacent to a component D in the graph $G - X$, then $w(G_1) = w(D) + w(s) \leq w(G - X) + w(s)$. Otherwise, s is an isolated vertex in $G - X'$, implying that $w(G_1) = w(s)$. We find that $w(G - X') = \max\{w(G_i) \mid 1 \leq i \leq r\} \leq w(G - X) + w(s)$. Consequently, $w(X') + w(G - X') \leq (w(X) - w(s)) + (w(G - X) + w(s)) = w(X) + w(G - X) = \iota(G)$, where the last equality follows from the assumption that X is an ι -set of G . We conclude that X' is an ι -set of G . \square

Given a graph G , the *incidence split graph* of G is the split graph $G^* = (C^*, I^*, E^*)$ consisting of a clique $C^* = \{v_x \mid x \in V(G)\}$ and an independent set $I^* = \{v_e \mid e \in E(G)\}$, where two vertices $v_x \in C^*$ and $v_e \in I^*$ are adjacent if and only if vertex x is incident with edge e in G . The following lemma will be used in hardness results not only in this section, but also in the section on component order connectivity (Section 3).

Lemma 2. *Let $G = (V, E)$ be a graph, $G^* = (C^*, I^*, E^*)$ its incidence split graph, and $k < |V|$ a non-negative integer. Then the following three statements are equivalent:*

- (i) G has a clique of size k ;
- (ii) there exists a set $X \subseteq C^*$ such that $|X| \leq k$ and $|X| + n(G^* - X) \leq |V| + |E| - \binom{k}{2}$;
- (iii) there exists a set $X \subseteq C^*$ such that $|X| \leq k$ and $n(G^* - X) \leq |V| + |E| - \binom{k}{2} - k$.

Proof. Let $n = |V|$ and $m = |E|$. We first prove that (i) implies (iii). Suppose G has a clique S of size k . Let $X = \{v_x \in C^* \mid x \in S\}$ denote the set of vertices in G^* corresponding to the vertices of S . Similarly, let $Y = \{v_e \in I^* \mid e \in E(G[S])\}$ denote the set of vertices in G^* corresponding to the edges in G both endpoints of which belong to S . Observe that $|Y| = \binom{k}{2}$ due to the fact that S is a clique of size k in G . Now consider the graph $G^* - X$. In this graph, every vertex of Y is an isolated vertex, while every vertex of $I^* \setminus Y$ has at least one neighbor in the clique $C^* \setminus X$. This implies that $n(G^* - X) = n + m - \binom{k}{2} - k$.

Since (iii) trivially implies (ii), it remains to show that (ii) implies (i). Suppose there exists a set $X \subseteq C^*$ such that $|X| \leq k$ and $|X| + n(G^* - X) \leq |V| + |E| - \binom{k}{2}$. Let $Z \subseteq I^*$ be the set of vertices in I^* both neighbors of which belong to X . Observe that $|Z| \leq \binom{|X|}{2}$ and $n(G^* - X) = n + m - |X| - |Z|$. Hence

$$n + m - \binom{k}{2} \geq |X| + n(G^* - X) = n + m - |Z| \geq n + m - \binom{|X|}{2},$$

which implies that $\binom{k}{2} \leq \binom{|X|}{2}$. Since $|X| \leq k$ by assumption, we find that $|X| = k$ and all the above inequalities must be equalities. In particular, we find that $|Z| = \binom{|X|}{2}$. We conclude that the vertices in G that correspond to X form a clique of size k in G . \square

Theorem 2. WEIGHTED VERTEX INTEGRITY is strongly NP-complete on split graphs.

Proof. We give a reduction from the NP-hard CLIQUE problem. Given an instance (G, k) of CLIQUE with $n = |V(G)|$ and $m = |E(G)|$, we create an instance (G', w, p) of WEIGHTED VERTEX INTEGRITY as follows. To construct G' , we start with the incidence split graph $G^* = (C^*, I^*, E^*)$ of G , and we add a single isolated vertex z . We define the weight function w by setting $w(z) = n + m - \binom{k}{2} - k$ and $w(v) = 1$ for every $v \in V(G') \setminus \{z\}$. Finally, we set $p = n + m - \binom{k}{2}$. For convenience, we assume that $k < n$.

We claim that G has a clique of size k if and only if $\iota(G') \leq p$. Since G' is a split graph and all the vertex weights are polynomial in n , this suffices to prove the theorem.

First suppose G has a clique S of size k . By Lemma 2, there exists a set $X \subseteq C$ such that $|X| \leq k$ and $n(G^* - X) \leq n + m - \binom{k}{2} - k$. Since $w(z) = n + m - \binom{k}{2} - k$ and every other vertex in G' has weight 1, it follows that $w(G' - X) = n + m - \binom{k}{2} - k$. Consequently, $w(X) + w(G' - X) \leq n + m - \binom{k}{2} = p$, so we conclude that $\iota(G') \leq p$.

For the reverse direction, suppose $\iota(G') \leq p$, and let $X \subseteq V(G')$ be an ι -set of G' . Due to Lemma 1, we may assume that $X \subseteq C$. We claim that $|X| \leq k$. For contradiction, suppose $|X| \geq k+1$. Then $w(X) = |X| \geq k+1$ and $w(G' - X) \geq w(z) = p - k$. This implies that $\iota(G') = w(X) + w(G' - X) \geq p+1$, yielding the desired contradiction. Now let D be the component of $G' - X$ containing the clique $C \setminus X$. The fact that every vertex in $V(G') \setminus \{z\}$ has weight 1 and the assumption that $k < n$ imply that $|D| = n(G^* - X) = w(G^* - X)$. Now observe that $|X| + n(G^* - X) \leq |X| + \max\{w(z), w(G^* - X)\} = \iota(G') \leq p = n + m - \binom{k}{2}$. We can therefore invoke Lemma 2 to conclude that G has a clique of size k . \square

The following result, previously obtained by Li et al. [16], is an easy consequence of Lemma 1. Theorem 4 below shows that this result is in some sense tight.

Theorem 3 ([16]). VERTEX INTEGRITY can be solved in linear time on split graphs.

Theorem 4. VERTEX INTEGRITY is NP-complete on chordal graphs.

Proof. We describe a slight modification of the reduction in the proof of Theorem 2. Given an instance (G, k) of CLIQUE, we construct a graph G'' in the same way as we constructed the graph G' , but instead of adding an isolated vertex z with weight $n + m - \binom{k}{2} - k$ in the last step, we add a clique of size $n + m - \binom{k}{2} - k$. We set $p = n + m - \binom{k}{2}$ as before. Using Lemma 2 and arguments similar to the ones in the proof of Theorem 2, it is not hard to show that the obtained instance (G'', p) of VERTEX INTEGRITY is a yes-instance if and only if (G, k) is a yes-instance of CLIQUE. The observation that G'' is a chordal graph completes the proof. \square

2.2 Parameterized Complexity

Recall that the VERTEX INTEGRITY problem is known to be fixed-parameter tractable when parameterized by p [6, 9]. To be more precise, Fellows and Stueckle [9] showed that the problem can be solved in time $O(p^{3p}n)$. A slight modification of their arguments yields the following result.

Theorem 5. WEIGHTED VERTEX INTEGRITY can be solved in $O(p^{p+1}n)$ time.

Proof. Let (G, w, p) be an instance of WEIGHTED VERTEX INTEGRITY, and let $n = |V(G)|$ and $m = |E(G)|$. We assume that every vertex in G has weight at least 1, as vertices of weight 0 can simply be deleted from the graph. This implies in particular that $|X| \leq w(X)$ for every set $X \subseteq V(G)$. We now show that we may also assume that $m \leq (p-1)n$. Suppose that (G, w, p) is a yes-instance. Then there exists a set $X \subseteq V(G)$ such that $w(X) + w(G - X) \leq p$. Let D_1, \dots, D_r be the components of $G - X$. Since every vertex has weight at least 1, it holds that $|X \cup D_i| \leq w(X \cup D_i) \leq p$ for each $i \in \{1, \dots, r\}$. Observe that G has a path decomposition of width at most $p-1$ whose bags are exactly

the sets $X \cup V(D_i)$. This implies that the pathwidth, and hence the treewidth, of G is at most $p - 1$. It is well-known that every n -vertex graph of treewidth at most t has at most tn edges [4]. We thus conclude that if (G, w, p) is a yes-instance, then $m \leq (p - 1)n$. Our algorithm can therefore safely reject the instance if $m > (p - 1)n$.

We now describe a simple branching algorithm that solves the problem. At each step of the algorithm, we use a depth-first search to find a set D of at most $p + 1$ vertices such that D induces a connected subgraph and $w(D) \geq p + 1$. If such a set does not exist, then every component of the graph under consideration has weight at most p , so the empty set is an ι -set of the graph and we are done. Otherwise, we know that any ι -set of the graph contains a vertex of D . We therefore branch into $|D| \leq p + 1$ subproblems: for every $v \in D$, we create the instance $(G - v, w, p - w(v))$, where we discard the instance in case $p - w(v) < 0$. Since the parameter p decreases by at least 1 at each branching step, the corresponding search tree T has depth at most p . Since T is a $p + 1$ -ary tree, it contains $O(p^p)$ nodes in total. Due to the assumption that $m \leq (p - 1)n$, the depth-first search at each step can be performed in time $O(pn)$. This yields an overall running time of $O(p^p pn) = O(p^{p+1}n)$. \square

We now show that the problem admits a polynomial kernel.

Theorem 6. *WEIGHTED VERTEX INTEGRITY admits a kernel with at most p^3 vertices, where each vertex has weight at most p .*

Proof. We describe a kernelization algorithm for the problem. Let (G, w, p) be an instance of WEIGHTED VERTEX INTEGRITY. We first delete all vertices of weight 0 without changing the parameter. Observe that after this preprocessing step, the weight of every vertex is at least 1, and hence $|X| \leq w(X)$ for every set $X \subseteq V(G)$. We apply the following two reduction rules.

Our first reduction rule starts by sorting the components of G according to their weights. Let D_1, \dots, D_r be the components of G such that $w(D_1) \geq w(D_2) \geq \dots \geq w(D_r)$. If $r > p + 1$, then we delete the component D_i for every $i \in \{p + 2, \dots, r\}$, without changing the parameter. In other words, we keep only the $p + 1$ heaviest components of G . Let G' be the obtained graph. To see why this rule is safe, it suffices to prove that (G, w, p) is a yes-instance if the new instance (G', w, p) is a yes-instance, as the reverse direction trivially holds. Suppose (G', w, p) is a yes-instance. Then there exists a set $X \subseteq V(G')$ such that $w(X) + w(G' - X) \leq p$. Since $|X| \leq w(X) \leq p$ and G' has exactly $p + 1$ components, there exists an index $i \in \{1, \dots, p + 1\}$ such that X does not contain any vertex from D_i . Since $w(D_i) \geq w(D_j)$ for every $j \in \{p + 2, \dots, r\}$, it holds that $w(G - X) = w(G' - X)$. Hence $w(X) + w(G - X) = w(X) + w(G' - X) \leq p$, implying that $\iota(G) \leq p$ and that (G, w, p) is a yes-instance.

The second reduction rule checks whether there exists a vertex $v \in V(G)$ for which $w(N_G[v]) > p$. Suppose such a vertex v exists. If $p - w(v) \geq 0$, then we delete v from the graph and reduce the parameter p by $w(v)$. If $p - w(v) < 0$, then we return a trivial no-instance. To see why this is safe, it suffices to show that if (G, w, p) is a yes-instance, then v belongs to any ι -set of G . Suppose (G, w, p) indeed is a yes-instance, and let X be an ι -set of G . Then $w(X) + w(G - X) = \iota(G) \leq p$. For contradiction, suppose that $v \notin X$. Consider the component D of $G - X$ that contains v . Since every vertex of $N_G[v]$ belongs either to X or to component D , and $w(N_G[v]) > p$ by assumption, we find that

$w(X) + w(D) > p$. But this implies that $w(X) + w(G - X) \geq w(X) + w(D) > p$, yielding the desired contradiction.

Let (G', w, p') denote the instance obtained after exhaustively applying the above reduction rules, where w denotes the restriction of the original weight function to the vertices of G' . Observe that $p' \leq p$. We assume that $p' \geq 2$, as otherwise we can trivially solve the instance (G', w, p') . We claim that if (G', w, p') is a yes-instance, then $|V(G')| \leq p^3$. Suppose (G', w, p') is a yes-instance, and let $X \subseteq V(G')$ be an ι -set of G' . Then $w(X) + w(D) \leq p' \leq p$ for every component D of $G' - X$. This, together with the fact that every vertex in G' has weight at least 1, implies that $|X| \leq p$ and $|D| \leq w(D) \leq p - w(X) \leq p - |X|$ for every component D of $G' - X$. Since the first reduction rule cannot be applied on the instance (G', w, p') , we know that G' has at most $p' + 1 \leq p + 1$ components. If $X = \emptyset$, then each of these components contains at most p vertices, so $|V(G')| \leq (p + 1)p \leq p^3$, where the last inequality follows from the assumption that $p \geq p' \geq 2$. Now suppose $|X| \geq 1$. Observe that every vertex in X has degree at most $p' \leq p$ due to the assumption that the second reduction rule cannot be applied. Hence, every vertex of X is adjacent to at most p components of $G' - X$, implying that there are at most p^2 components of $G' - X$ that are adjacent to X . Since G' itself has at most $p + 1$ components, at least one of which contains a vertex of X , we find that $G' - X$ has at most $p^2 + p$ components in total. Recall that each of these components contains at most $p - |X|$ vertices. We conclude that $|V(G')| \leq (p^2 + p)(p - |X|) + |X| \leq p^3$, where we use the assumption that $|X| \geq 1$. Due to the second reduction rule, each vertex in G' has weight at most p .

It remains to argue that our kernelization algorithm runs in polynomial time. Observe that the execution of any reduction rule strictly decreases either the number of vertices in the graph or the parameter, so each rule is applied only a polynomial number of times. The observation that each rule can be executed in polynomial time completes the proof. \square

3 Component Order Connectivity

In this section, we turn our attention to the COMPONENT ORDER CONNECTIVITY problem and its weighted variant. After proving some NP-hardness results in Section 3.1, we present two algorithms for interval graphs in Section 3.2. Section 3.3 contains our parameterized and kernelization complexity results on the two problems.

3.1 Classical Complexity: Hardness Results

It is easy to see that (G, p) is a yes-instance of VERTEX INTEGRITY if and only if there exist non-negative integers k and ℓ with $k + \ell = p$ such that (G, k, ℓ) is a yes-instance of COMPONENT ORDER CONNECTIVITY. Hence, any instance (G, p) of VERTEX INTEGRITY can be solved by making at most p calls to an algorithm solving COMPONENT ORDER CONNECTIVITY, yielding the following observation:

Observation 1 *There is a polynomial-time Turing reduction from VERTEX INTEGRITY to COMPONENT ORDER CONNECTIVITY.*

Due to the above observation, it holds that COMPONENT ORDER CONNECTIVITY is NP-complete on any graph class for which VERTEX INTEGRITY is NP-complete.

Our next two results identify graph classes for which WEIGHTED COMPONENT ORDER CONNECTIVITY and COMPONENT ORDER CONNECTIVITY are strictly harder than WEIGHTED VERTEX INTEGRITY and VERTEX INTEGRITY, respectively.

Theorem 7. *WEIGHTED COMPONENT ORDER CONNECTIVITY is NP-complete on complete graphs.*

Proof. We reduce from PARTITION, which is the problem of determining whether a multiset A of positive integers can be partitioned into two subsets A_1 and A_2 such that the sum of the elements in A_1 equals the sum of the elements in A_2 . This problem is well-known to be NP-complete [10]. Given an instance $A = \{a_1, \dots, a_n\}$ of PARTITION, we construct an instance (G, w, k, ℓ) of WEIGHTED COMPONENT ORDER CONNECTIVITY as follows. We define G to be a complete graph with vertex set $V = \{v_1, \dots, v_n\}$, and the weight function w is defined by setting $w(v_i) = a_i$ for every $i \in \{1, \dots, n\}$. Let $W = 1/2 \sum_{i=1}^n a_i$. We set $k = \ell = W$.

Suppose A can be partitioned into two subsets A_1 and A_2 such that the sum of the elements in A_i equals W for each $i \in \{1, 2\}$. Let $V_1 = \{v_i \in V \mid a_i \in A_1\}$ be the subset of vertices of G corresponding to the set A_1 . Since G is a complete graph and $w(V) = 2W$, is clear that $w(V_1) = W$ and $w(G - V_1) = W$, implying that (G, w, k, ℓ) is a yes-instance of WEIGHTED COMPONENT ORDER CONNECTIVITY. The reverse direction is similar: if there exists a subset $V' \subseteq V$ with $w(V') = W$ and $w(G - V') = W$, then the partition $V', V \setminus V'$ of V corresponds to a desired partition A_1, A_2 of A . \square

Recall that the VERTEX INTEGRITY problem is linear-time solvable on split graphs by Theorem 3.

Theorem 8. *COMPONENT ORDER CONNECTIVITY is NP-complete on split graphs.*

Proof. We give a reduction from the NP-hard CLIQUE problem. Let (G, k) be an instance of CLIQUE with $n = |V(G)|$ and $m = |E(G)|$. Let $G^* = (C^*, I^*, E^*)$ be the split incidence graph of G , and let $\ell = n + m - \binom{k}{2}$. By Lemma 2, there is a clique of size k in G if and only if there exists a set $X \subseteq C^*$ such that $|X| \leq k$ and $n(G^* - X) \leq n + m - \binom{k}{2}$. This immediately implies that (G, k) is a yes-instance of CLIQUE if and only if (G^*, k, ℓ) is a yes-instance of COMPONENT ORDER CONNECTIVITY. \square

Observe that WEIGHTED COMPONENT ORDER CONNECTIVITY cannot be solved in polynomial time on interval graphs due to Theorem 7, unless $P = NP$. Similarly, a pseudo-polynomial algorithm for WEIGHTED COMPONENT ORDER CONNECTIVITY on split graphs is unlikely to exist, since the problem is strongly NP-complete on that class as a result of Theorem 2 and Observation 1. In Section 3.2 we present a pseudo-polynomial algorithm for WEIGHTED COMPONENT ORDER CONNECTIVITY on interval graphs, as well as an $O(n^2)$ -time algorithm for COMPONENT ORDER CONNECTIVITY on the same graph class.

3.2 Classical Complexity: Algorithms for Interval Graphs

In this section, we present a pseudo-polynomial algorithm, called wCOC, that solves the WEIGHTED COMPONENT ORDER CONNECTIVITY problem in $O(kn^3)$ time on interval

graphs. In Theorem 9, we show how this algorithm can be slightly modified in such a way that it solves the same problem in $O(\ell n^3)$ time on interval graphs. Finally, we present yet another modification of the algorithm that solves the unweighted version of the problem on interval graphs in $O(n^2)$ time.

We now give an informal description of the algorithm wCOC; we refer to Figure 1 for pseudocode of the algorithm. Given an instance (G, w, k, ℓ) , where G is an interval graph, the algorithm first removes every vertex of weight 0. It then computes a *clique path* of G , i.e., an ordering K_1, \dots, K_t of the maximal cliques of G such that for every vertex $v \in V(G)$, the maximal cliques containing v appear consecutively in this ordering. Since G is an interval graph, such an ordering exists and can be obtained in $O(n^2)$ time [5]. For convenience, we define two empty sets K_0 and K_{t+1} . The algorithm now computes the set $S_i = K_i \cap K_{i+1}$ for every $i \in \{0, \dots, t\}$. Observe that S_0 and S_t are both empty by construction, and that the non-empty sets among S_1, \dots, S_{t-1} are exactly the minimal separators of G (see, e.g., [12]). For every $q \in \{0, \dots, t+1\}$, we define $G_q = G[\cup_{i=0}^q K_i]$. Also, for any two integers i, j with $0 \leq i < j \leq t$, the algorithm computes the set $V_{i,j} = V(G[\cup_{p=i+1}^j K_p \setminus (S_i \cup S_j)])$. Informally speaking, the set $V_{i,j}$ consists of the vertices of G that lie “in between” separators S_i and S_j .

Let us give some intuition behind the next phase of the algorithm. Suppose (G, w, k, ℓ) is a yes-instance of WEIGHTED COMPONENT ORDER CONNECTIVITY, and let X be a solution for this instance. Generally speaking, X fully contains some minimal separators of G whose removal is necessary to break the graph into pieces, as well as additional vertices that are deleted from these pieces with the sole purpose of decreasing the weight of each piece to at most ℓ . The constructed clique path K_1, \dots, K_t corresponds to a linear order of the minimal separators S_1, \dots, S_{t-1} of G . We will use this linear structure to find a minimum solution by doing dynamic programming over the minimal separators of G .

For every $q \in \{0, \dots, t\}$, let k_q denote the smallest integer such that there exists a set $X \subseteq V(G)$ satisfying the following three properties:

- $w(X) = k_q$;
- S_q is a subset of X ;
- X is a solution for the instance (G_q, w, k_q, ℓ) .

In other words, X is a “cheapest” solution for (G_q, w, k_q, ℓ) that fully contains the minimal separator S_q . The algorithm now constructs an array dp with $t+1$ entries, each of which is an integer from $\{0, \dots, k+1\}$. Initially, all the elements of the array are set to $k+1$. For any $q \in \{0, \dots, t\}$, we say that the entry $\text{dp}[q]$ has *reached optimality* if

$$\text{dp}[q] = \begin{cases} k_q & \text{if } k_q \leq k \\ k+1 & \text{otherwise.} \end{cases}$$

Recall that $S_t = \emptyset$ and that $G_t = G$. Hence, if $\text{dp}[t]$ has reached optimality, then the input instance (G, w, k, ℓ) is a yes-instance if and only if $\text{dp}[t] \leq k$. In Lemma 3, we will prove the correctness of the algorithm by showing that $\text{dp}[t]$ has indeed reached optimality by the end of the algorithm.

The algorithm uses a subroutine **MinSup** that, given a multiset of weights $\{w_1, \dots, w_n\}$ and a target W such that $\sum_{i=1}^n w_i \geq W$, finds a set $I \subseteq \{1, \dots, n\}$ such that $\sum_{i \in I} w_i$ is minimized with respect to the constraint $\sum_{i \in I} w_i \geq W$. Note that this subroutine **MinSup**

can be implemented to run in time $O(Wn)$ using a trivial extension of the classical dynamic programming algorithm for SUBSET SUM.

Algorithm **wCOC**

Input: An instance (G, w, k, ℓ) of WEIGHTED COMPONENT ORDER CONNECTIVITY, where G is an interval graph

Output: “yes” if (G, w, k, ℓ) is a yes-instance, and “no” otherwise

Remove every vertex of weight 0 from G
Construct K_0, \dots, K_{t+1}
Construct S_0, \dots, S_t
Construct $V_{i,j}$ for every $0 \leq i < j \leq t$

Set all elements of dp to $k + 1$
Set $\text{dp}[0] = 0$

for j from 1 to t **do**

for i from $j - 1$ to 0 **do**

Let $v_1, \dots, v_{|V_{i,j}|}$ be the vertices of $V_{i,j}$
Let $w_p = w(v_p)$ for every $p \in \{1, \dots, |V_{i,j}|\}$
if $w(V_{i,j}) \leq k + \ell$ **then**

Let $I = \text{MinSup}(\{w_1, \dots, w_{|V_{i,j}|}\}, w(V_{i,j}) - \ell)$
Let $Y_{i,j} = \{v_p \in V_{i,j} \mid p \in I\}$
 $\text{dp}[j] = \min \begin{cases} \text{dp}[j] \\ \text{dp}[i] + w(Y_{i,j}) + w(S_j \setminus S_i) \end{cases}$

end

end

end

return “yes” if $\text{dp}[t] \leq k$, and “no” otherwise

Fig. 1. Pseudocode of the algorithm **wCOC** that solves the WEIGHTED COMPONENT ORDER CONNECTIVITY problem on interval graphs in $O(kn^3)$ time.

Lemma 3. *Given an instance (G, w, k, ℓ) of WEIGHTED COMPONENT ORDER CONNECTIVITY, where G is an interval graph, the algorithm **wCOC** returns “yes” if and only if (G, w, k, ℓ) is a yes-instance.*

Proof. Recall that in order to prove the lemma, it suffices to prove that by the end of the algorithm, $\text{dp}[t]$ has reached optimality. For each $j \in \{1, \dots, t\}$, we define $P_1(j)$ to be the statement “at the start of iteration j of the outer loop, $\text{dp}[i]$ has reached optimality for every $i < j$,” and we define $P_2(j)$ to be the statement “at the end of iteration j of the outer loop, $\text{dp}[i]$ has reached optimality for every $i \leq j$.” Observe that the following claim trivially holds.

Claim 1. *For any $q \in \{1, \dots, t - 1\}$, it holds that $P_2(q)$ implies $P_1(q + 1)$.*

We now prove another claim.

Claim 2. For any $q \in \{1, \dots, t\}$, it holds that $P_1(q)$ implies $P_2(q)$.

In order to prove Claim 2, we first prove that if $\text{dp}[q] \leq k$ at the end of iteration q , then there is a solution of weight $\text{dp}[q]$ for the instance $(G_q, w, \text{dp}[q], \ell)$ that contains S_q . Let $r < q$ be such that $\text{dp}[q] = \text{dp}[r] + w(Y_{r,q}) + w(S_q \setminus S_r)$. Due to our initialization of the table dp and the assumption that $\text{dp}[q] \leq k$, such an r exists. Because we assume that $P_1(q)$ holds, $\text{dp}[r]$ has reached optimality. Hence, there is a set $X_r \subseteq V(G_r)$ such that $w(X_r) = \text{dp}[r]$, X_r contains S_r , and X_r is a solution for $(G_r, w, \text{dp}[r], \ell)$. Consider the set $Y_{r,q}$ that was constructed using the subroutine **MinSup**. Recall that $w(Y_{r,q}) \geq w(V_{r,q}) - \ell$ and hence $w(V_{r,q}) \leq w(Y_{r,q}) + \ell$. Furthermore, by assumption, the only component of $G_q - (X_r \cup S_q)$ possibly of weight larger than ℓ is a component of $G[V_{r,q}]$. Let $X_q = X_r \cup S_q \cup Y_{r,q}$. Due to the correctness of **MinSup**, it holds that any component of $G_q - X_q$ is of weight at most ℓ . Hence X_q is a solution for $(G_q, w, \text{dp}[q], \ell)$ that contains S_q and has weight $\text{dp}[q]$.

It remains to prove that for any set $X \subseteq V(G_q)$ such that $S_q \subseteq X$ and $w(G_q - X) \leq \ell$, it holds that $w(X) \geq \text{dp}[q]$. Assume, for contradiction, that there exists a set $X \subseteq V(G_q)$ such that $S_q \subseteq X$, $w(G_q - X) \leq \ell$, and $w(X) < \text{dp}[q]$. Let $r < q$ be the largest index such that $S_r \subseteq X$. Observe that r exists due to the fact that $S_0 = \emptyset$. We claim that

$$w(V_{r,q} \setminus X) > \ell.$$

First consider the case where $w(V_{r,q}) > k + \ell$. Then $w(V_{r,q} \setminus X) \geq w(V_{r,q}) - w(X) > k + \ell - w(X) \geq \ell$, where the last inequality follows from the fact that $w(X) < \text{dp}[q] \leq k + 1$.

Now consider the case where $w(V_{r,q}) \leq k + \ell$. Then by the algorithm, $\text{dp}[q] \leq \text{dp}[r] + w(Y_{r,q}) + w(S_q \setminus S_r)$. By definition, we know that the sets $V(G_r)$, $V_{r,q}$, and $S_q \setminus S_r$ form a partition of $V(G_q)$, and hence $w(X \cap V(G_r)) + w(X \cap V_{r,q}) + w(X \cap (S_q \setminus S_r)) = w(X) < \text{dp}[q] \leq \text{dp}[r] + w(Y_{r,q}) + w(S_q \setminus S_r)$. Since $S_r \subseteq X$ and $P_1(q)$ holds by assumption, we have that $\text{dp}[r] \leq w(X \cap V(G_r))$. From $S_q \subseteq X$ it follows that $w(X \cap (S_q \setminus S_r)) = w(S_q \setminus S_r)$ and since we know that $w(X \cap V(G_r)) \geq \text{dp}[r]$ it follows immediately that $w(X \cap V_{r,q}) < w(Y_{r,q})$. By the correctness of **MinSup** it follows that $w(X \cap V_{r,q}) < w(V_{r,q}) - \ell$. Hence, we find that $w(V_{r,q} \setminus X) = w(V_{r,q}) - w(X \cap V_{r,q}) > w(V_{r,q}) - (w(V_{r,q}) - \ell) = \ell$.

We now prove that $G[V_{r,q} \setminus X]$ is connected. Let u and v be two distinct vertices from $V_{r,q} \setminus X$. We will now prove that u and v belong to the same component of $G[V_{r,q} \setminus X]$. If there is a maximal clique K_i of G containing both u and v , then this trivially holds. Suppose this is not the case. Let a be the largest number such that $u \in K_a$ and b the smallest number such that $v \in K_b$. Assume without loss of generality that $a < b$. By construction of $V_{r,q}$, it follows that $r < a < b \leq q$. By the definition of r , we know that $S_z \setminus X$ is non-empty for every $z \in [a, b)$. Consequently, the vertices in $\cup_{z \in [a, b)} S_z$ induce a connected subgraph of $G[V_{r,q} \setminus X]$, and u and v are thus contained in the same component of this graph.

Recall that $w(G_q - X) \leq \ell$ by the definition of X . However, $G[V_{r,q} \setminus X]$ is a component of $G_q - X$ that has weight $w(V_{r,q} \setminus X) > \ell$. This yields the desired contradiction, and completes the proof of Claim 2.

We now show that $P_1(1)$ holds. In order to show this, it suffices to argue that $\text{dp}[0]$ has reached optimality at the start of the first iteration of the outer loop. Recall that $\text{dp}[0]$ is set to 0 during the initialization phase, so $\text{dp}[0]$ equals 0 at the start of the first iteration of the outer loop. Since $S_0 = V(G_0) = \emptyset$, it holds that $k_0 = 0$.

Since $P_1(1)$ holds, we can repeatedly apply Claims 1 and 2 to deduce that $P_2(t)$ holds. Hence, $\text{dp}[t]$ has reached optimality by the end of the last iteration of the outer loop. \square

Theorem 9. WEIGHTED COMPONENT ORDER CONNECTIVITY can be solved in $O(sn^3)$ time on interval graphs, where $s = \min\{k, \ell\}$.

Proof. Due to Lemma 3, it suffices to prove that the algorithm wCOC runs in time $O(sn^3)$, where $s = \min\{k, \ell\}$. Clearly, we can remove all vertices of weight 0 in $O(n^2)$ time. It is well-known that a clique path of an interval graph can be constructed in $O(n^2)$ time, and that an interval graph has no more than n maximal cliques [13]. Consequently, all the sets K_0, \dots, K_{t+1} and S_0, \dots, S_t can be constructed in $O(n^2)$ time. Observe that for all $0 \leq i < j \leq t$, it holds that $V_{i,j} = V_{i,j-1} \cup (V(K_j) \setminus (S_i \cup S_j))$. Hence, once the sets K_0, \dots, K_{t+1} and S_0, \dots, S_t have been constructed, the sets $V_{i,j}$ can be computed in $O(n^3)$ time using a straightforward dynamic programming procedure.

We claim that the body of inner loop runs in time $O(kn)$. Observe that the body of this loop is only executed if $w(V_{i,j}) \leq k + \ell$. Since $|V_{i,j}| \leq n$ and $w(V_{i,j}) - \ell \leq k + \ell - \ell = k$, the algorithm MinSup solves the instance $(\{w_1, \dots, w_{|V_{i,j}|}\}, w(V_{i,j}) - \ell)$ in $O(kn)$ time, which is therefore also the time it takes to obtain $Y_{i,j}$. Clearly, the value of $\text{dp}[j]$ can be computed in $O(n)$ time. Since the inner loop is executed $O(n^2)$ times, we conclude that wCOC terminates in time $O(kn^3)$.

It remains to argue why WEIGHTED COMPONENT ORDER CONNECTIVITY can be solved in time $O(\ell n^3)$ in case $\ell < k$. Recall the following two lines from the inner loop of the algorithm wCOC, explaining how we obtain the set $Y_{i,j}$:

Let $I = \text{MinSup}(\{w_1, \dots, w_{|V_{i,j}|}\}, w(V_{i,j}) - \ell)$
 Let $Y_{i,j} = \{v_p \in V_{i,j} \mid p \in I\}$

The idea is to replace the subroutine MinSup by a subroutine MaxInf that, given a multiset of weights $\{w_1, \dots, w_n\}$ and a target W , finds a set $I \subseteq \{1, \dots, n\}$ such that $\sum_{i \in I} w_i$ is maximized under the constraint $\sum_{i \in I} w_i \leq W$. It is clear that MaxInf, just like MinSup, can be solved in $O(Wn)$ time. By replacing the above two lines in the inner loop by the following two lines, we can obtain the exact same set $Y_{i,j}$ in $O(\ell n)$ time:

Let $I = \text{MaxInf}(\{w_1, \dots, w_{|V_{i,j}|}\}, \ell)$
 Let $Y_{i,j} = \{v_p \in V_{i,j} \mid p \notin I\}$

This slight modification yields an algorithm for solving WEIGHTED COMPONENT ORDER CONNECTIVITY on interval graphs in $O(\ell n^3)$ time. \square

Theorem 10. COMPONENT ORDER CONNECTIVITY can be solved in $O(n^2)$ time on interval graphs.

Proof. We describe a modification of the algorithm wCOC, called uCOC, that solves the unweighted COMPONENT ORDER CONNECTIVITY problem in $O(n^2)$ time on interval graphs. There are two reasons why the algorithm wCOC does not run in $O(n^2)$ time: constructing all the sets $V_{i,j}$ takes $O(n^3)$ time in total, and each of the $O(n^2)$ executions of the inner loop takes $O(kn)$ time, which is the time taken by the subroutine MinSup to compute the set $Y_{i,j}$ of vertices that are to be deleted.

Recall that for every $j \in \{1, \dots, t\}$ and every $i \in \{0, \dots, j-1\}$, the set $Y_{i,j}$ computed by the algorithm wCOC is defined to be the minimum-weight subset of $V_{i,j}$ for which the weight of the subgraph $G[V_{i,j}] - Y_{i,j}$ is at most ℓ . Also recall that once the set $Y_{i,j}$ is computed, the value of $\text{dp}[j]$ is updated as follows:

$$\text{dp}[j] = \min \begin{cases} \text{dp}[j] \\ \text{dp}[i] + w(Y_{i,j}) + w(S_j \setminus S_i) \end{cases}$$

When solving the unweighted variant of the problem, we can decrease the weight (i.e., order) of the subgraph $G[V_{i,j}]$ to at most ℓ by simply deleting $|V_{i,j}| - \ell$ vertices from $V_{i,j}$ in a greedy manner. In other words, it is no longer important to decide *which* vertices to delete from $V_{i,j}$, but only *how many* vertices to delete. This means that we can replace the entire body of the inner loop by the following line:

$$\text{dp}[j] = \min \begin{cases} \text{dp}[j] \\ \text{dp}[i] + (|V_{i,j}| - \ell) + |S_j \setminus S_i| \end{cases}$$

Hence, in order to finish the proof of the lemma, it suffices to argue that we can precompute the values $|V_{i,j}|$ and $|S_j \setminus S_i|$ for every $j \in \{1, \dots, t\}$ and $i \in \{0, \dots, j-1\}$ in $O(n^2)$ time in total.

Recall that $V_{i,j} = V(G[\cup_{p=i+1}^j K_p \setminus (S_i \cup S_j)])$ by definition, so

$$|V_{i,j}| = |\cup_{p=i+1}^j K_p| - |S_i| - |S_j| + |S_i \cap S_j|.$$

Moreover, it is clear that

$$|S_j \setminus S_i| = |S_j| - |S_i \cap S_j|.$$

The algorithm **uCOC** starts by computing the sets K_0, \dots, K_{t+1} and S_0, \dots, S_t as before in $O(n^2)$ time, as well as the cardinalities of these sets. For each $v \in V(G)$, let $L(v)$ denote the largest index i such that $v \in K_i$. Observe that we can compute the value $L(v)$ for all $v \in V(G)$ in $O(n^2)$ time in total. The algorithm then computes the value $|\cup_{p=0}^i K_p|$ for every $i \in \{0, \dots, t\}$. Using these values, it then computes the value $|\cup_{p=i+1}^j K_p| = |\cup_{q=0}^j K_q| - |\cup_{r=0}^i K_r| + |K_i \cap K_{i+1}| = |\cup_{q=0}^j K_q| - |\cup_{r=0}^i K_r| + |S_i|$ for every $j \in \{1, \dots, t\}$ and every $i \in \{0, \dots, j-1\}$. Observe that this can also be done in $O(n^2)$ time in total.

It remains to show that we can compute the value $|S_i \cap S_j|$ for all indices i and j with $0 \leq i < j \leq t$ in $O(n^2)$ time in total. Let us fix an index $i \in \{0, \dots, t\}$. Since we precomputed the L -value of each vertex and we can order the vertices in S_i by increasing L -value in $O(n)$ time, we can compute the value $|S_i \cap S_j| = |\{v \in S_i \mid L(v) \geq j+1\}|$ for all $j \in \{i, \dots, t\}$ in $O(n)$ time in total. This completes the proof. \square

3.3 Parameterized Complexity

In this section, we investigate the parameterized complexity and kernelization complexity of (WEIGHTED) COMPONENT ORDER CONNECTIVITY. Since COMPONENT ORDER CONNECTIVITY is equivalent to VERTEX COVER when $\ell = 1$, this problem is para-NP-hard when parameterized by ℓ . The fact that the reduction in the proof of Theorem 8 is parameter-preserving implies that the unweighted version of the problem is also hard when parameterized by k :

Corollary 1. *COMPONENT ORDER CONNECTIVITY is $W[1]$ -hard when parameterized by k , even when restricted to split graphs.*

On the positive side, our next result shows that the problem becomes fixed-parameter tractable when parameterized by $k + \ell$, even when we allow arbitrary vertex weights. Given the similarities between vertex integrity and component order connectivity, it comes as no surprise that we will be able to re-use arguments from the previous section. However, since there are some subtle differences between the proofs in this section and the ones in Section 2.2, we will give all the details here in order to improve the readability of this section.

Theorem 11. WEIGHTED COMPONENT ORDER CONNECTIVITY *can be solved in time $O(\ell^k(k + \ell)n) = 2^{O(k \log \ell)}n$.*

Proof. Let (G, w, k, ℓ) be an instance of WEIGHTED COMPONENT ORDER CONNECTIVITY, and let $n = |V(G)|$ and $m = |E(G)|$. We assume that every vertex in G has weight at least 1, as vertices of weight 0 can simply be deleted from the graph. Suppose that (G, w, k, ℓ) is a yes-instance. Then there exists a set $X \subseteq V(G)$ such that $w(X) \leq k$ and $w(G - X) \leq \ell$. Let D_1, \dots, D_r be the components of $G - X$. We can construct a path decomposition of G by taking as bags the sets $X \cup D_i$ for all $i \in \{1, \dots, r\}$. Since every vertex has weight at least 1, we know that each bag contains at most $k + \ell$ vertices, implying that G has treewidth at most $k + \ell - 1$. Consequently, G has at most $(k + \ell - 1)n$ edges due to a result by Bodlaender and Fomin [4]. We may therefore assume that $m \leq (k + \ell - 1)n$, as our algorithm can safely reject the instance otherwise.

We now describe a simple branching algorithm that solves the problem. Now, at each step of the algorithm, we use a depth-first search to find a set D of at most $\ell + 1$ vertices such that $w(D) \geq \ell + 1$ and D induces a connected subgraph. If such a set does not exist, then every component of the graph has weight at most ℓ , so we are done. Otherwise, we know that any solution contains a vertex of D . We therefore branch into $|D| \leq \ell + 1$ subproblems: for every $v \in D$, we create the instance $(G - v, w, k - w(v), \ell)$, where we discard the instance in case $k - w(v) < 0$. Since the parameter k decreases by at least 1 at each branching step, the corresponding search tree T has depth at most k . Since T is an $(\ell + 1)$ -ary tree of depth at most k , it has at most $((\ell + 1)^{k+1} - 1)/((\ell + 1) - 1) = O(\ell^k)$ nodes. Due to the assumption that $m \leq (k + \ell - 1)n$, the depth-first search at each step can be performed in time $O(n + m) = O((k + \ell)n)$. This yields an overall running time of $O(\ell^k(k + \ell)n) = 2^{O(k \log \ell)}n$. \square

We now show that the branching algorithm in Theorem 11 is in some sense best possible. In order to make this statement concrete, we need to introduce some additional terminology.

For $k \geq 3$, let s_k be the infimum of the set of all positive real numbers δ for which there exists an algorithm that solves k -SAT in time $O(2^{\delta n})$, where n denotes the number of variables in the input formula. The Exponential Time Hypothesis (ETH) states that $s_k > 0$ for any $k \geq 3$ [14]. In particular, this implies that there is no $2^{o(n)}$ -time algorithm for solving 3-SAT, unless the ETH fails. Lokshtanov, Marx, and Saurabh [17] developed a framework for proving lower bounds on the running time of parameterized algorithms for certain natural problems, assuming the validity of the ETH. In order to obtain these results, they proved lower bounds for constrained variants of some basic problems such as the following:

$k \times k$ CLIQUE

Instance: A graph G , and a partition \mathcal{X} of $V(G)$ into k sets X_1, \dots, X_k of size k each.

Question: Does G have a clique K such that $|K \cap X_i| = 1$ for all $i \in \{1, \dots, k\}$?

Theorem 12 ([17]). *There is no $2^{o(k \log k)}$ time algorithm for $k \times k$ CLIQUE, unless the ETH fails.*

Recall that the WEIGHTED COMPONENT ORDER CONNECTIVITY problem can be solved in time $2^{O(k \log \ell)} n$ on general graphs. We now show that the problem does not admit a $2^{o(k \log \ell)} n^{O(1)}$ -time algorithm, even when all the vertices have weight 1 and the input graph is a split graph, unless the ETH fails.

Theorem 13. *There is no $2^{o(k \log \ell)} n^{O(1)}$ time algorithm for COMPONENT ORDER CONNECTIVITY, even when restricted to split graphs, unless the ETH fails.*

Proof. For contradiction, suppose there exists an algorithm \mathbb{A} for solving the COMPONENT ORDER CONNECTIVITY problem in time $2^{o(k \log \ell)} n^{O(1)}$. Let (G, \mathcal{X}) be an instance of the $k \times k$ CLIQUE problem, where $\mathcal{X} = \{X_1, \dots, X_k\}$. We assume that G contains no edge whose endpoints belong to the same set X_i , as an equivalent instance can be obtained by deleting all such edges from G . Due to this assumption, it holds that (G, \mathcal{X}) is a yes-instance of $k \times k$ CLIQUE if and only if G contains a clique of size k .

Now let $G^* = (C^*, I^*, E^*)$ be the incidence split graph of G , and let $\ell = |V(G)| + |E(G)| - \binom{k}{2}$. By the definition of the $k \times k$ CLIQUE problem, we have that $|V(G)| = k^2$ and $|E(G)| \leq k^2(k^2 - 1)/2$. This implies that the graph G^* has at most $k^2 + k^2(k^2 - 1)/2 \leq k^4$ vertices, and that $\ell \leq k^4$. By Lemma 2, it holds that (G^*, k, ℓ) is a yes-instance of COMPONENT ORDER CONNECTIVITY if and only if G has a clique of size k . Hence, using algorithm \mathbb{A} , we can decide in time $2^{o(k \log k^4)} k^{O(1)} = 2^{o(k \log k)}$ whether or not (G, \mathcal{X}) is a yes-instance of $k \times k$ CLIQUE. This contradicts Theorem 12. \square

We conclude this section by showing that the WEIGHTED COMPONENT ORDER CONNECTIVITY problem admits a polynomial kernel.

Theorem 14. *WEIGHTED COMPONENT ORDER CONNECTIVITY admits a kernel with at most $k\ell(k + \ell) + k$ vertices, where each vertex has weight at most $k + \ell$.*

Proof. We describe a kernelization algorithm for the problem. Let (G, w, k, ℓ) be an instance of WEIGHTED COMPONENT ORDER CONNECTIVITY. We first delete all vertices of weight 0 without changing the parameters. Observe that after this first preprocessing step, the weight of every vertex is at least 1. This implies in particular that $|X| \leq w(X)$ for every set $X \subseteq V(G)$.

We now apply the following two reduction rules. If G contains a vertex v such that $w(N_G[v]) > k + \ell$, then we delete v from G and decrease k by $w(v)$, unless $w(v) > k$, in which case we output a trivial no-instance. To see why this rule is safe, let us first show that v belongs to any solution for the instance (G, w, k, ℓ) if such a solution exists. This follows from the observation that deleting any set $X \subseteq V(G) \setminus \{v\}$ with $w(X) \leq k$ from G yields a graph G' such that $w(N_{G'}[v]) > \ell$. For the same reason, there exists no solution if $w(v) > k$. Our second reduction rule deletes any component D of weight at most ℓ from G without changing either of the parameters. This rule is safe due to the fact that $w(D) \leq \ell$ implies that no minimum solution deletes any vertex from D .

Let (G', w, k', ℓ) denote the instance that we obtain after exhaustively applying the above reduction rules, where w denotes the restriction of the original weight function to the vertices of G' . Observe that $k' \leq k$, while the parameter ℓ did not change in the kernelization process. Suppose X is a solution for this instance. Then $w(X) \leq k' \leq k$, which implies that X contains at most k vertices. For every component D of $G' - X$, it holds that $|D| \leq w(D) \leq \ell$. Moreover, every component D of $G' - X$ is adjacent to at least one vertex of X , as otherwise our second reduction rule could have been applied. Moreover, the fact that the first reduction rule cannot be applied guarantees that $w(N_G[v]) \leq k + \ell$ for every $v \in V(G')$. In particular, this implies that every vertex in X has degree at most $k + \ell$. We find that $G - X$ has at most $k(k + \ell)$ components, each containing at most ℓ vertices. We conclude that if (G', w, k', ℓ') is a yes-instance, then $|V(G')| \leq k\ell(k + \ell) + k$. The observation that each vertex in G' has weight at most $k + \ell$ due to the first reduction rule completes the proof. \square

4 Concluding Remarks

Our NP-completeness result for VERTEX INTEGRITY on co-comparability graphs, together with the polynomial-time algorithm for WEIGHTED VERTEX INTEGRITY by Ray et al. [19] on interval graphs, raises the question whether WEIGHTED VERTEX INTEGRITY can be solved in polynomial time on permutation graphs. Recall that the unweighted version of this problem can be solved in $O(n^5)$ time [15] on this graph class.

We showed that the COMPONENT ORDER CONNECTIVITY problem does not admit a $2^{o(k \log \ell)} n^{O(1)}$ time algorithm, unless the ETH fails. Can the problem be solved in time $c^{k+\ell} n^{O(1)}$ for some constant c ? Similarly, it would be interesting to investigate whether it is possible to solve VERTEX INTEGRITY in time $c^p n^{O(1)}$ for some constant c .

References

1. Bagga, K.S., Beineke, L.W., Goddard, W., Lipman, M.J., Pippert, R.E.: A survey of integrity. *Discrete Applied Mathematics* 37, 13–28 (1992)
2. Barefoot, C.A., Entringer, R., Swart, H.: Vulnerability in graphs—a comparative survey. *J. Combin. Math. Combin. Comput* 1(38), 13–22 (1987)
3. Ben-Ameur, W., Mohamed-Sidi, M.A., Neto, J.: The k -separator problem. In: Du, D.Z., Zhang, G. (eds.) *COCOON. Lecture Notes in Computer Science*, vol. 7936, pp. 337–348. Springer (2013)
4. Bodlaender, H.L., Fomin, F.V.: Equitable colorings of bounded treewidth graphs. *Theor. Comput. Sci.* 349(1), 22–30 (2005)
5. Booth, K.S., Lueker, G.S.: Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *J. Comput. Syst. Sci.* 13(3), 335–379 (1976)
6. Clark, L.H., Entringer, R.C., Fellows, M.R.: Computational complexity of integrity. *J. Combin. Math. Combin. Comput* 2, 179–191 (1987)
7. Diestel, R.: *Graph Theory (Graduate Texts in Mathematics)*. Springer (2005)
8. Downey, R.G., Fellows, M.R.: *Parameterized Complexity*. Springer-Verlag (1999), 530 pp.
9. Fellows, M., Stueckle, S.: The immersion order, forbidden subgraphs and the complexity of network integrity. *J. Combin. Math. Combin. Comput* 6, 23–32 (1989)
10. Garey, M.R., Johnson, D.S.: *Computers and Intractability*. W. H. Freeman, New York (1979)

11. Gross, D., Heinig, M., Iswara, L., Kazmierczak, L.W., Luttrell, K., Saccoman, J.T., Suffel, C.: A survey of component order connectivity models of graph theoretic networks. *WSEAS Transactions on Mathematics* 12, 895–910 (2013)
12. Ho, C.W., Lee, R.C.T.: Counting clique trees and computing perfect elimination schemes in parallel. *Information Processing Letters* 31(2), 61–68 (1989)
13. Ibarra, L.: The clique-separator graph for chordal graphs. *Discrete Applied Mathematics* 157(8), 1737 – 1749 (2009)
14. Impagliazzo, R., Paturi, R.: Complexity of k -SAT. In: *IEEE Conference on Computational Complexity*. pp. 237–240. IEEE Computer Society (1999)
15. Kratsch, D., Kloks, T., Müller, H.: Measuring the vulnerability for classes of intersection graphs. *Discrete Applied Mathematics* 77(3), 259–270 (1997)
16. Li, Y., Zhang, S., Zhang, Q.: Vulnerability parameters of split graphs. *Int. J. Comput. Math.* 85(1), 19–23 (2008)
17. Lokshtanov, D., Marx, D., Saurabh, S.: Slightly superexponential parameterized problems. In: Randall, D. (ed.) *SODA*. pp. 760–776. SIAM (2011)
18. Ray, S., Deogun, J.S.: Computational complexity of weighted integrity. *J. Combin. Math. Combin. Comput.* 16, 65–73 (1994)
19. Ray, S., Kannan, R., Zhang, D., Jiang, H.: The weighted integrity problem is polynomial for interval graphs. *Ars Combinatoria* 79, 77–95 (2006)
20. Woeginger, G.J.: The toughness of split graphs. *Discrete Mathematics* 190(1-3), 295–297 (1998)